UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/676,373	09/30/2003	Stefan Jesse	09700.0216-00	3224

60668 7590 01/09/2009

SAP / FINNEGAN, HENDERSON LLP
901 NEW YORK AVENUE, NW
WASHINGTON, DC 20001-4413

EXAMINER	
VU, TUAN A	

ART UNIT	PAPER NUMBER
2193	

MAIL DATE	DELIVERY MODE
01/09/2009	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/676,373	JESSE ET AL.	
	Examiner	Art Unit	
	TUAN A. VU	2193	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *10 October 2008*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-4,6-12 and 14-24* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-4,6-12 and 14-24* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. This action is responsive to the Applicant's response filed 10/10/08.

As indicated in Applicant's response, claims 1, 10, 18 have been amended, and claims 23-24 added. Claims 1-4, 6-12, 14-24 are pending in the office action.

Double Patenting

2. The nonstatutory double patenting rejection is based on a judicially created doctrine grounded in public policy (a policy reflected in the statute) so as to prevent the unjustified or improper timewise extension of the "right to exclude" granted by a patent and to prevent possible harassment by multiple assignees. A nonstatutory obviousness-type double patenting rejection is appropriate where the conflicting claims are not identical, but at least one examined application claim is not patentably distinct from the reference claim(s) because the examined application claim is either anticipated by, or would have been obvious over, the reference claim(s). See, e.g., *In re Berg*, 140 F.3d 1428, 46 USPQ2d 1226 (Fed. Cir. 1998); *In re Goodman*, 11 F.3d 1046, 29 USPQ2d 2010 (Fed. Cir. 1993); *In re Longi*, 759 F.2d 887, 225 USPQ 645 (Fed. Cir. 1985); *In re Van Ornum*, 686 F.2d 937, 214 USPQ 761 (CCPA 1982); *In re Vogel*, 422 F.2d 438, 164 USPQ 619 (CCPA 1970); and *In re Thorington*, 418 F.2d 528, 163 USPQ 644 (CCPA 1969).

A timely filed terminal disclaimer in compliance with 37 CFR 1.321(c) or 1.321(d) may be used to overcome an actual or provisional rejection based on a nonstatutory double patenting ground provided the conflicting application or patent either is shown to be commonly owned with this application, or claims an invention made as a result of activities undertaken within the scope of a joint research agreement.

Effective January 1, 1994, a registered attorney or agent of record may sign a terminal disclaimer. A terminal disclaimer signed by the assignee must fully comply with 37 CFR 3.73(b).

3. Claims 8, 18 are provisionally rejected on the ground of nonstatutory obviousness-type double patenting as being unpatentable over claims 4, 12, 19 of copending Application No. 10,676,374 (hereinafter '374).

Although the conflicting claims are not identical, they are not patentably distinct from each other because of the following example of conflicting claims.

As per instant claim 8, copending '374 claim 4 recites a first data model being used to derive an API and employing the API to access development objects. But '374 claim 4 does not explicitly recite (i) first model defining development objects as building blocks for the application; (ii) generate intermediate objects therefrom, intermediate objects comprising Java objects, and (iii) using the set of intermediate objects as inputs to generate the API, the model including (iv) a first model customized extension used to implement a feature of the API such as an indication of a file border and (v) API enforces relationships and constraints defined in the first model.

However, '374 claim 4 recites a variation of the language in claim 8 for limitations (i) and (ii) via the recital of 'defining file borders comprising identifying of development objects to be included in a file … in the data model … associated component class … to be children of the main …object that are not identified as main…objects', the intermediate objects being added objects to the file of the main object including development objects being OO classes defined from the data model; rendering the teaching in claim 4 obvious language variation of (i) and (ii). As for the *constraints enforcing* limitation of (v) based (iii)-- *using the set of intermediate objects as input for the API generating*—'374 claim 4 includes file storing user-defined code associated with the main development object; and in view '374 teaching of a definition file (see '374 claim 4) in light of objects being defined -- in terms of parent/child relationship in '374 claim 3 – one of ordinary skill would recognize these defined objects as well-known interrelated model components viewed in a GUI development interface of '374, i.e. API - to necessarily support user's development via instantiating one such development GUI API for accessing model objects. And this as a whole would be equivalent to (iii) for enforcing constraints as

suggested above, or otherwise obvious variation thereof. As for the *customizable extension* comprising a file border indication referred to as (iv), this is suggested in '374 reciting of 'defining file borders for development', and storing development objects in a repository based on the file borders, and accessing these objects via the API (*); so that one skill in the art would be motivated to provide an extension structure obtained from the repository (e.g. template builder) in the course of the API derivation with utilizing of information in the '374 stored file-based repository for the derivation. That is, the information thus extended (e.g. via a template builder) from the stored model/repository regarding a particular file border identity would be used to support the creation of API parameter or attributes which would be needed to access the very components stored from the '374 defining of file borders, as purported by the endeavor described as (*) from the above.

As per instant claim 8, '374 claim 19 also recites API derived from a data model, file borders defined in the model, and user interface using the API to access development objects being stored in a repository. Claim 19 does not recite 'first language model with extension to implement API and file border; but based on '374 reciting of association between component and model class 'that associates a user interface to a ... application model', the extension by use of border file suggest the *extension* limitation of instant claim 8 to provide deriving of association between stored model objects; that is, obviousness in terms of instant claim 8 limitations such as API for 'enforcing constraints' and 'language extension' does apply here in view of the rationale set forth above.

As per instant claim 18, '374 claim 12 also recites an obvious language variant thereof in expressing 'receiving of a model' in a development method, a *language extension* for defining

a model representing blocks in terms of 'component class' and 'model class' as well as their inter-association for developing an application (Note: this would be an obvious variation of building block relationships among objects and, hence suggesting constraints thereof), deriving a API based thereon, and use the API for *enforcing constraints* in the model within the development of the application.

This is a provisional obviousness-type double patenting rejection because the conflicting claims have not in fact been patented.

Claim Rejections - 35 USC § 103

4. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

5. Claims 1-4, 6-9, 18-24 are rejected under 35 U.S.C. 103(a) as being unpatentable over Charisius et al., USPN: 2002/0108101, (hereinafter Charisius)

As per claim 1, Charisius discloses a computer program product, tangibly embodied in an information carrier, for developing an application, the computer program product being operable to cause data processing apparatus to:

receive a first model in a first language (e.g. Fig. 9, 10A,B; para 0090, pg. 13 - Note: files defining object classes and retrieved to build a OO model in terms of UML representations – see Unified Modeling Language - para 0088-0089, pg. 13 -- reads on first model in first language – see Fig. 11 – such as Java or C++ language) from a storage device, the first model defining

development objects representing building blocks for developing the application (e.g. Fig. 12-18), relationships among developing objects, and constraints for developing the application (see *generalized definitions for the particular language ... Use Case Modeling* - para 0089, pg. 13 - Note: definition from retrieved file definition in accordance to standardized UML notations and language rules reads on constraints between UML objects);

generate a set of intermediate objects using the first model (e.g. para 0066, pg. 4; para 0093-0096, pg. 13-14; Fig. 12-19) wherein the set of intermediate objects comprises Java objects (Fig. 12-19); and

using the set of intermediate objects, generate an API (e.g. para 0092-0096, pg. 13 – Note: using the packages from definition source files along with graphical *view* of class symbols represented in UML model, code objects to generate an metamodel within interface 610, including instance of RWI, IDE or SCI **reads on** API being instantiated using intermediate objects, i.e. an instance of *static, dynamic, or functional* view - see para 0092 - within Interface 610 for further tasks; para 0109, pg. 15 – Note: using a modifiable and dynamic instance of the XML structure diagram to build correspondence between a DTD model and target XML structure reads on input for a parser component of above API – see Fig. 7 – the parser instance or API needed for parsing intermediate objects or derived markup language -- see Fig. 26B-C; parser– see Fig. 24; step 2116, Fig. 21B) wherein the API enables accessing the development objects (e.g. Fig. 7; DE: *extract information from the model* – para 0065; *access information* – para 0068, pg. 4; para 0089-0090, pg. 13).

Charisius does not explicitly disclose that the instance of API generated from the first model enforces the relationships and constraints defined in the first model. But Charisius

teaches using a graphical representation to visualize relationships between OO elements in the UML model views (e.g. *use case, time ordering, structural organization, sequences of states* - para 0092-0094, pg. 13), and along with the using of additional modules to support the created API (see modules 704 -Fig. 7 i.e. support RWI, IDE or SCI module), the concept of graphically determining whether the derived (classes) objects **are compliant** with their being structured or logical sequence dictated from their hierarchical relationship or timing, input/output implications (see *use case, time ordering, structural organization, sequences of states … messages ... collaboration ... desired operation ... emphasize time ordering* – para 0092-0094, pg. 13) is suggestive of a form of enforcing relationships as these are observed on the views (see Fig.12-19). It would have been obvious for one skill in the art at the time the invention was made to implement the Quality Assurance modules (para 0068, pg. 4) and the IDE support modules (Fig. 7) along with the UML-derived graphical view instances of the TMM (see para 0092, pg. 13; Figure 3-6) to enforce relationships and constraints defined in the 'first model' as set forth above, because of the very nature of object relationship defined and regulated as UML and class inheritance due to the object-oriented nature of source or templatized classes as endeavored by Charisius use of IDE via support of the above auditing capabilities when generating OO objects in a editable and updatable viewer to achieve a correct state of a TMM (see Fig. 10 and 11) viewed within the interface 610 as purported by Charisius development cycle (see Figure 2 and Figure 7)

As per claim 2, Charisius discloses wherein the set of intermediate objects is generated using a second data model (e.g. Fig. 21A-C and related text; para 0105, pg. 14) and instructions

to convert the first model to a second data model in a second language(para 0099, pg. 14; *parses graphical view file ... for generating the XML structure* - para 0121, pg. 17; step 2112, Fig. 26B).

As per claim 3, Charisius discloses that the second language model comprises XML (see para 0099, pg. 14).

As per claim 4, Charisius disclose wherein the first language is UML (e.g. para 0088-0089, pg. 13).

As per claims 6-8, Charisius discloses wherein the first language comprises a customizable extension (e.g. view 204, TMM 200, code editor 208 -Fig. 2; para 0064-0067, pg. 4); wherein the customizable extension is used to implement an additional feature of the API (e.g. para 0064-0067, pg. 4; Fig. 9), wherein the additional feature comprises an indication of a file border (e.g. *file is new... file ... been updated* -- para 0090, pg. 13– Note: repository of model – Fig. 2-5 --having files being enlisted for a project and identified for its update status **reads on** model extension with indication to file borders including management or versioning metric).

As per claim 9, Charisius discloses wherein the API comprises a copy and paste operation (e.g. Fig. 12-18, 22, 23; para 0090-0094, pg. 13 – Note: customization via user interface (see GUI pane with toolbar) to create instance of API from the core API of Fig. 7 whereby the IDE enables modeling and delete/add of development objects reads on GUI API in which *copy and paste* are features operable user's modifications of a UML view).

As per claim 18, Charisius discloses a computer program product, tangibly embodied in an information carrier, for developing an application, the computer program product being operable to cause data processing apparatus to:

receive a data model defining development objects … developing the application from a storage device, relationships among … objects, and constraints for developing application, wherein the development objects comprise Java objects (refer to claim 1) ;

derive an API based on the data model ... in the data model (refer to claim 1); and

use the API to perform operations on the development objects (e.g. refer to claim 1).

But Charisius does not explicitly disclose that the instance of API generated from the data model enforces the relationships and constraints defined in the data model. The API enforcing limitation has been addressed as obvious in claim 1.

As per claim 19, Charisius discloses wherein the API comprises an interface layer (e.g. para 0064-0067, pg. 4 Note: RWI API reads on interface layer wherein diagrams can be user driven), a proxy layer (e.g. IDE API reads on proxy layer wherein information are channeled, extracted and filtered for the interface layer to used) a state layer (e.g. SCI API reads on state layer wherein data received as-is is just displayed for plain view and editing by the RWI).

As per claim 20, Charisius discloses wherein the operations comprise creating a new development object as a transient object (e.g. Fig. 12-18); and modifying the transient object until the transient object is committed to a persistent file (e.g. Fig. 12-18; ; para 0090-0094, pg. 13).

As per claims 21-22, Charisius discloses comprising instructions to destroy the transient object if a delete command is requested before the transient object is committed to a persistent file; and to mark the persistent file as deleted if a delete is requested after the transient object is committed to a persistent file (e.g. Fig. 2; Fig. 7; Fig. 10AB; Fig 24-26; para 0118-0119, pg. 17 –

Note: use of TMM transient structure to enable modifying/removing – as in not marked for persisting or committed for file repository – using the created API when parsing DTD or XML data which are previously stored **reads on** modifying a transient object and generate code when such object is committed; while retrieving corresponding DTD/XML reads on reusable objects being committed for DB persistence from a previous development instance).

As per claims 23-24, Charisius discloses storage module (e.g. *location 2306* - para 0107, pg. 15 - for storing definition files) as storage device.

Claim Rejections - 35 USC § 102

6. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

7. Claims 10-12, 14-17 are rejected under 35 U.S.C. 102(b) as being anticipated by Charisius et al., USPN: 2002/0108101, (hereinafter Charisius).

As per claim 10, Charisius discloses a computer program product, tangibly embodied in an information carrier, for developing an application, the computer program product being operable to cause data processing apparatus to:

receive a first model in a first language from a storage device (refer to claim 1) … relationships among the development objects (para 0101, pg. 14) and constraints for developing the application (refer to claim 1), wherein the first language comprises unified modeling language (para 0088-0089, pg. 13);

generate (a set of intermediate objects …) using the first model (para 0066, pg. 4; para 0093-0096, pg. 13-14; Fig. 12-19);

and generate an XML schema (e.g. Fig. 21-24; *parses graphical view file ... for generating the XML structure* - para 0121, pg. 17; step 2612 Fig. 26B) using the set of intermediate objects as inputs such that the schema (Note: W3C methodology reads on *schema* language with integral syntactic rules enforcing of relationship among hierarchy of markup elements) enforces the relationships and constraints defined in the first model (refer to claims 1, 2) so to enable implementation of development objects.

As per claim 11, the use of a XML intermediate structure (e.g. para 0109, pg. 1) as a temporary XML model representation to incorporate more intermediate OO data constructs (e.g. see Fig. 25, 26) reads on using a second model to dynamically generate more objects.

As per claim 12 refer to the corresponding rejection as set forth in claim 3.

As per claim 14, see (e.g. Java class – para 0089, pg. 13)

As per claims 15-16, Charisius discloses wherein the XML schema includes a tree based on aggregation relationships in the first data model; wherein the XML schema includes a reference based on an association relationship in the first data model (e.g. Fig. 24-25; Figs 26).

As per claim 17, Charisius discloses wherein the XML schema includes a complex type extension based on an inheritance relationship in the first data model (e.g. Fig. 12-18; *JAVA, group... defining elements, "hierarchy"*, para 0124-0126, pg. 18 – Note: UML and Java constructs parsed with construction of AC3 DTD and XML hierarchy reads on inheritance within some complex type in which a group is linked to constituting subelements – see Fig. 25).

Response to Arguments

8. Applicant's arguments filed 10/10/08 have been fully considered but they are not persuasive. Following are the Examiner's observation in regard thereto.

35 USC § 103 Rejection:

(A) Applicants have submitted that Charisius fails to teach or suggest 'receiving a first model in a first language from a storage device' (Appl. Rmrks pg. 9, middle). The Office Action has been updated to match the added limitation; and the 'from a storage device' is deemed fulfilled with Charisius retrieving of OO object definition files purport to implement a model, hence has disclosed 'first model in a first language' as file format.

(B) Applicants have submitted that Charisius does not teach or suggest (Appl. Rmrks pg. 10, bottom) generating based on graphical view representation, but rather from a source code. The Office Action has matched intermediate objects being Java classes assembled together (in form of UML views) from model file definition (i.e. first model in first language). The Applicants' mentioning of 'based on graphical view representation' is not commensurate with the exact language (with respect to ;generating … intermediate objects') of claim 1 by any stretch of imagination. Applicant's arguments fail to comply with 37 CFR 1.111(b) because they amount to a general allegation that the claims define a patentable invention without specifically pointing out how the language of the claims patentably distinguishes them from the reference.

(C) Applicants have submitted that UML model, interface 610, RWI, IDE, SCI were alleged by the Office Action to read on API as claimed (Appl. Rmrks pg. 11 bottom, pg. 12 top) and this is not what Charisius discloses, because Charisius' IDE, RWI, SCI compose the development tool not generating an API. The Office Action has made it clear that instance of a API for

developing a model starting from reading definition files and assembling UML construct as graphical views amount to generating a API; that is, the API is the graphical interface being put together as a application programming instance for the developer to further expand the UML views based on the initial model definition files being read to form intermediate OO objects (i.e. Java components in UML format). The Argument is not sufficient to overcome the rejection because "API" as claimed has not been detailed sufficiently to otherwise enforce a more narrow interpretation. Applicant's arguments fail to comply with 37 CFR 1.111(b) because they amount to a general allegation that the claims define a patentable invention without specifically pointing out how the language of the claims patentably distinguishes them from the reference.

35 USC § 102 Rejection:

(D) Applicants have submitted that for anticipation, receiving XML documents as proffered in the Office Action does not constitute receiving a model in a first language (Appl. Rmrks, pg. 14, top). The XML file is deemed a metamodel containing hierarchy of meta-elements that serve as input into a modeling framework or as output therefrom, because of its very multi-platform portability. The Office Action has made it clear (refer to claim 1) that first model in first language (first language being UML type definition) are for example OO object definition files; and that deriving views of UML objects reads on intermediate objects. Based on these UML views, structure diagram in XML can be formed (see para 0121, pg. 17; refer to claim 2). The argument is not persuasive.

In all, the claims stand rejected as set forth in the Office Action.

Conclusion

9. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Tuan A Vu whose telephone number is (571) 272-3735. The examiner can normally be reached on 8AM-4:30PM/Mon-Fri.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Lewis Bullock can be reached on (571)272-3759.

The fax phone number for the organization where this application or proceeding is assigned is (571) 273-3735 (for non-official correspondence - please consult Examiner before using) or 571-273-8300 (for official correspondence) or redirected to customer service at 571-272-3609.

Any inquiry of a general nature or relating to the status of this application should be directed to the TC 2100 Group receptionist: 571-272-2100.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

/Tuan A Vu/

Primary Examiner, Art Unit 2193

January 07, 2009